A perfect sampling algorithm of random walks with forbidden arcs

Stéphane Durand, Bruno Gaujal, Florence Perronnin, Jean-Marc Vincent

A perfect sampling algorithm of random walks with forbidden arcs.

Stéphane Durand , Bruno Gaujal , Florence Perronnin , Jean-Marc Vincent

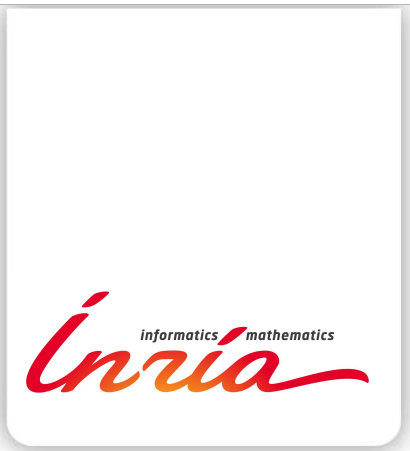
RESEARCH REPORT

N° 8504

Mar. 2014

Project-Team Mescal

ISSN 0249-6399 ISRN INRIA/RR--8504--FR+ENG



A perfect sampling algorithm of random walks with forbidden arcs.

Stéphane Durand [*] [†], Bruno Gaujal [†] [‡] [§], Florence Perronnin [‡] [§] [†], Jean-Marc Vincent [‡] [§] [†]

Project-Team Mescal

Research Report n° 8504 — Mar. 2014 — 23 pages

Abstract: In this paper we show how to construct an algorithm to sample the stationary distribution of a random walk over $\{1,\dots,N\}^d$ with forbidden arcs. This algorithm combines the rejection method and coupling from the past of a set of trajectories of the Markov chain that generalizes the classical sandwich approach. We also provide a complexity analysis of this approach in several cases showing a coupling time in $O(N^2 d \log d)$ when no arc is forbidden and an experimental study of its performance.

Key-words: Perfect simulation, Markov chain, random walks

[*] ENS of Lyon, France
[†] Inria
[‡] Univ. Grenoble Alpes, LIG, F-38000 Grenoble, France
[§] CNRS, LIG, F-38000 Grenoble, France

RESEARCH CENTRE GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Algorithme de simulation parfaite des Marches aléatoires avec pas interdits

Résumé : Dans cet article, nous montrons comment construire un algorithme de simulation parfaite d'une marche aléatoire sur $\{1, \ldots, N\}^d$ avec certains pas interdits. Cet algorithme combine une méthode du rejet avec la technique du couplage depuis le passé pour un ensemble de trajectoire qui généralise l'approche classique par enveloppes. Nous faisons aussi une analyse de la complexité de cette approche et nous montrons en particulier que dans le cas sans pas interdits, le temps de couplage de l'algorithme est en $O(N^2 d \log d)$. Une campagne expérimentale de tests est aussi fournie pour faire une évaluation pratique des performances en temps et en mémoire de l'algorithme.

Mots-clés : Simulation parfaite, chaîne de Markov, marche aléatoire

1 Introduction

Random walks are rampant in the literature on Markov chains (see for example [Aldous and Fill, 2002]). Several fundamental questions in Markov chains such as *hitting time* (the first time a state is reached) and the *mixing time* (the time it takes for the measure of a chain to be close to its stationary measure) are particularly acute for random walks.

In this paper, we consider the problem of sampling from the stationary distribution of a random walk over a multidimensional grid. Up to our knowledge, this problem has not been studied before, neither at the theoretical nor at the algorithmic level.

This problem can be approached using Monte-Carlo simulation, that only converges asymptotically. Here we rather use a *perfect sampling* approach that provides a perfect sample in finite time. The efficiency of Monte-Carlo simulation is given by the mixing time of the chain. However, for perfect sampling, the time complexity is given by the *coupling time* of the chain (the duration until two coupled chains starting from any two states, coalesce). Up to our knowledge, coupling times (or coalescence times) of random walks have not been studied before and must be evaluated to estimate the number of steps needed by a perfect sampler.

Furthermore, two additional difficulties have to be solved to design an effective perfect sampling algorithm that can handle large state spaces.

The first difficulty is the design of a constructive definition of the Markov chain under the form $X_{n+1} = \phi(X_n, U_{n+1})$ (where U_{n+1} is a uniform random variable on $[0, 1]$) or in other words, finding a *grand coupling* of the random walk, using the terminology of [Levin et al., 2008]. We propose one solution in Section 3.1 where the discrete time random walk is transformed into a continuous time Markov chain, for which a grand coupling can be constructed. The bias introduced by this transformation is removed using a rejection method.

The second difficulty comes from the fact that the random walk (discrete or continuous) with forbidden arcs is not monotone so that classical perfect sampling techniques based on the simulation of extreme points ([Propp and Wilson, 1996]) is not valid here. Furthermore, specific techniques for non-monotone chains based on upper and lower bounds of all trajectories (often called the *sandwich* method [Kendall and Møller, 2000]) are not possible here either, because such bounds will not coalesce in most cases. This second problem is solved by introducing a more sophisticated version of the bounds/split paradigm of [Busic et al., 2008], presented in Section 3.3. We construct a tight superset of all trajectories of the random walks starting from all possible states. This superset is made of a set of isolated points and one interval (given by its two extreme points). While the number of isolated points increases during the simulation, we show that this increase remains moderate (logarithmic in the size of the state space) and does not hamper the applicability of the algorithm even for large state spaces, at least when the number of forbidden arcs remains reasonably small (see the complexity Section 4).

Finally, we are able to analyse the time and space complexity of our Algorithm. Its time complexity (equal to the coalescence time of the chain, up to a factor 4) is shown to be logarithmic in the size of the state space when without forbidden arcs. Experimental results show that this complexity remains of the same order when forbidden arcs are added. We also make sure that the space complexity of the algorithm remains linear in the number of forbidden arcs.

Here is the structure of the paper. Section 2 shows several domains where random walks in high dimensions are useful objects. Section 3 presents the construction of the sampling algorithm based on two original ideas: first we explain how rejection can be used in this context, then we introduced the generalization of sandwich simulation using intervals and isolated points. In Section 4 we analyze the time and space complexity of our algorithm and we report experimental studies in Section 5.

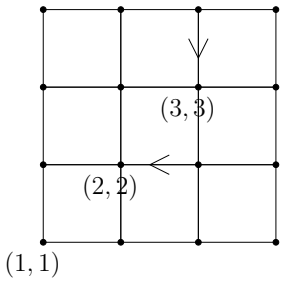


Figure 1: random walk over $\{1,2,3,4\}^2$, with two forbidden arcs. The forbidden arcs are displayed with arrows like one-way roads on roadmaps. Here a move to the right from point $(2,2)$ is forbidden as well as a move up from point $(3,3)$.

2 Random walks in $\mathbb{Z}^d$

Let us consider a random walk over a finite grid in dimension d, $\mathcal{S} \stackrel{\text{def}}{=} \{1,\ldots,N\}^d$ where both the *span* N of the grid and its *dimension* d are large.

The set $\mathcal{S}$ is equipped with the componentwise order: For any $\mathbf{x},\mathbf{y} \in \mathcal{S}$, $\mathbf{x} \preccurlyeq \mathbf{y}$ if $x_i \leqslant y_i$ for all $i = 1,..,d$. For this order, the smallest point is $\mathbf{1} = (1,1,\ldots,1)$ and the largest point is $\mathbf{N} = (N,...,N)$. The *interval* $[\mathbf{a},\mathbf{b}]$ denotes the set of points larger than $\mathbf{a}$ and smaller than $\mathbf{b}$: $[\mathbf{a},\mathbf{b}] \stackrel{\text{def}}{=} \{\mathbf{x} \in \mathcal{S}, s.t. \mathbf{a} \preccurlyeq \mathbf{x} \preccurlyeq \mathbf{b}\}$. Let $\mathbf{e}_i \stackrel{\text{def}}{=} (0,...,1,...,0)$ with 1 in the i-th position. From position $\mathbf{x}$, the walker can move uniformly to $\mathbf{x} \pm \mathbf{e}_i$ unless this is a forbidden arc or gets him out of the grid.The position of the walker after n steps is denoted $X(n)$ and forms a discrete time Markov chain with state space $\mathcal{S}$. An example in dimension 2 is displayed in Figure 1.

The position of the walker after n steps is denoted $X(n)$ and forms a discrete time Markov chain with state space $\mathcal{S}$. If the state space is strongly connected (we always assume this is true in the rest of paper), this random walk has a unique stationary measure ν. This paper is devoted to the construction of an efficient algorithm that provides a point in $\mathcal{S}$ distributed according to the stationary measure of $X(n)$.

It should be clear that as soon as the random walk contains one forbidden arc, it is not reversible in general and computing its stationary measure ν becomes difficult. Up to our knowledge, the best way the compute ν is then to solve the balance equations $\nu = \nu P$. The complexity of this approach is in $O(N^{3d})$ in time and $O(N^{2d})$ in space, unusable even for reasonable values of N and d. Here, we develop an algorithm that samples according to ν in logarithmic time and space.

2.1 Potential Applications

Random walks have be used in several domains, ranging from statistical physics to models of parallel systems.

2.1.1 Percolation over a finite graph

Random walks with forbidden arcs are well studied in the literature. They are the main object in percolation theory studied intensively by mathematicians and physicists since the early seminal work [Broadbent and Hammersley, 1957].

In general, in percolation theory, the forbidden edges are picked at random (arcs in both directions on the edge are forbidden), each with probability p. Note that in that case, the random walk is a reversible Markov chain. Percolation in directed graph also exist [Schwartz et al., 2002].

Here, we study the case where the state space is finite so the classical question of escape to infinity is not relevant. Instead we analyze the effect of the forbidden arcs on the coupling time

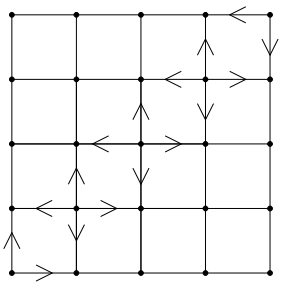


Figure 2: Two particles moving a discrete finite line as a random walk in dimension 2. Forbidden arcs prevent the walker (the two particles) to reach the diagonal (occupy the same position).

of the chain, and therefore on the difficulty to sample the stationary distribution.

2.1.2 Interacting particles

The random walk in dimension $d = DM$ can also be used to model the movements of M particles moving in a space of dimension D, with asynchronous movements of the particles. The random walk then consists in the concatenation of the coordinates of all particles. In that context, the forbidden arcs correspond to the fact that certain types of movements of some particles in space, are not allowed due to the current positions of the other particles.

The example of two particles moving on the discrete line (dimension 1) that cannot occupy the same location can be seen as a Markov chain in dimension 2, with forbidden arcs preventing the walker to reach the diagonal (see Figure 2). Of course, in this particular example the two particles cannot cross each other so the Markov chain is not irreducible. However, as soon as the particles evolve in dimension D higher than 1, this problem disappears. In the remainder of the paper we will always assume that the Markov chain is irreducible. The forbidden states on diagonal hyperplanes can simply be removed from the state space to ensure irreducibility.

Our analysis (see Section 4) will show that the sampling time of the corresponding Markov chain grows in the number of particles in an acceptable way ($O(M \log M)$) in the case with no forbidden arcs. While this may be unacceptable to study a system made of a very large number of particles, we believe this approach is adapted to the study of a moderate number of particles whose interactions are very complex.

2.1.3 Stochastic Automata Networks

Stochastic Automata Networks (SANs) have been shown to be a powerful approach to model parallel systems with a small degree of synchronization [Plateau and Stewart, 1997]. In the simple case where each automata is a birth and death process and synchronizations correspond to blocking transitions, SANs are random walks with forbidden arcs as studied here, where forbidden arcs correspond to synchronization constraints.

As an illustration, let us consider the famous dining philosophers. This is a random walk where the dimension d is the number of philosophers and the span is $\{0, 1\}$ (states of one philosopher). The arcs from $(a_1, \ldots, a_{i-1}, 0, a_{i+1}, \ldots, a_d)$ to $(a_1, \ldots, a_{i-1}, 1, a_{i+1}, \ldots, a_d)$ are forbidden if a_{i-1} or a_{i+1} is equal to one.

3 Perfect Sampling Algorithm

This section is devoted to the construction of a perfect sampling algorithm of a random walk $X(n)$ over $\mathcal{S}$ where certain arcs are forbidden.

This construction is done in several steps and the final algorithm is provided in pseudo-code (Algorithm 1).

3.1 Grand coupling and rejection

The random walk over a grid with forbidden arcs is an irreducible, finite, discrete time Markov chain over a finite state space $\mathcal{S}$ denoted $X(n)_{n\in\mathbb{N}}$, with transition matrix P. By definition, for any position $\mathbf{a}$ and any direction $\mathbf{m} = \pm\mathbf{e}_i$, $P_{\mathbf{a},\mathbf{a}+\mathbf{m}} = \frac{1}{q_\mathbf{a}}$ where $q_\mathbf{a}$ is the number of possible moves from $\mathbf{a}$.

From $X(n)_{n\in\mathbb{N}}$, one can construct a continuous time Markov chain $Y(t)_{t\in\mathbb{R}}$ over the same state space. The generator Q of Y is obtained by multiplying each line $\mathbf{a}$ in P by $q_\mathbf{a}$ and defining the diagonal element $Q_{\mathbf{a},\mathbf{a}}$ as $Q_{\mathbf{aa}} = -\sum_\mathbf{b} q_\mathbf{a} P_\mathbf{ab}$. Therefore, the rates from $\mathbf{a}$ to $\mathbf{a}+\mathbf{m}$ are all equal to one: $Q_{\mathbf{a},\mathbf{a}+\mathbf{m}} = 1$.

From $Y(t)$, it is possible to extract a new discrete time Markov chain, $Y(n)_{n\in\mathbb{N}}$ by uniformization. Its transition matrix is $Id + \Lambda^{-1}Q$, where Λ (uniformization constant) is any positive real number larger than all $q_\mathbf{a}$'s. Since the total rate out of any state in Y is bounded by $2d$, it can be uniformized by $\Lambda = 2d$.

While it can be difficult to construct a grand coupling for chain X, such a construction is easy and natural for the chain Y since the rates are all equal. To couple the walks starting from all states, just pick one direction uniformly among the $2d$ possibilities and make every walk move in that direction. Those for which the move is not possible stay still.

This yields the following constructive definition of the chain $Y(n)$ under the form $Y(n+1) = \phi(Y(n), \mathbf{m}_{n+1})$ with the following definition for the function ϕ.

Definition 1 (of constructive function ϕ)**.** *In any state $\mathbf{y} \in \mathcal{S}$, and for any direction $\mathbf{m}$ among the $2d$ possibilities $\pm\mathbf{e}_1, \ldots, \pm\mathbf{e}_d$ (chosen uniformly) then*

$$\phi(\mathbf{y}, \mathbf{m}) = \begin{cases} \mathbf{y} + \mathbf{m} & \text{if the move is valid} \\ \mathbf{y} & \text{otherwise.} \end{cases}$$

This coupling makes the chain Y more attractive for perfect sampling. Unfortunately, the chain Y and the initial chain X do not have the same stationary probability distribution. However it is possible to construct a procedure that generates a stationary sample of the initial chain X, from a stationary sample of the chain Y. This procedure (given in Algorithm 1) is based on a rejection method.

Let us consider that the construction of $Y(n)$ is given by

$$Y(n+1) = \phi(Y(n), \mathbf{m}_{n+1}),$$

where ϕ is the deterministic function given above and $\mathbf{m}_{n+1}$ a uniform random variable among $\{\pm\mathbf{e}_1, \ldots, \pm\mathbf{e}_d\}$. If Y can be sampled efficiently (using an algorithm $PSA(Y)$ that will be described later) then X can also be sampled efficiently provided rejection is unlikely.

Theorem 1. *If the algorithm $PSA(Y)$ samples Y under its stationary distribution, then the rejection Algorithm $PSA(X)$ gives a sample distributed according to the stationary distribution of $X(n)$.*

Proof. Let us call μ_i the stationary probability of state i for the uniformized (as well as for the continuous time) chain Y and ν_i the the stationary probability of state i for original Markov chain $X(n)$. By construction, the original Markov chain $X(n)$ is the embedded chain at jump

Algorithm 1: Sampling algorithm of $X(n)$ ($PSA(X)$)

Data: Algorithm $PSA(Y)$ sampling Y perfectly; a random move $\mathbf{m}_1$.
Result: A state sampled from the stationary distribution of X

```
begin
    repeat
        Sample Y(0) with stationary distribution (using Algo. PSA(Y))
        Simulate one step: Y(1) := φ(Y(0), m_1);
    until Y(1) ≠ Y(0)                    //reject if Y(1) = Y(0);
    return Y(0)
```

times of the time continuous version $Y(t)$. It is well known [Brémaud, 1998] that $q_i\mu_i \propto \nu_i$ where q_i is the rate out of state i. Let us compute the probability that the output of the rejection algorithm is i. Let T be the first time when the sample $Y(0)_T$ is not rejected:

$$\mathbb{P}(Y(0)_T = i) = \sum_{t=0}^{\infty} \mathbb{P}(T = t, Y(0)_t = i) = \sum_{t=0}^{\infty} R^{t-1}\mu_i \frac{q_i}{\Lambda},$$

where the probability of rejection R of the t-th sample does not depend on t and is equal to $R = \sum_j \mu_j \frac{\Lambda - q_j}{\Lambda}$. This implies that $\mathbb{P}(Y(0)_T = i) = \frac{1}{1-R}\mu_i \frac{q_i}{\Lambda}$.

Therefore, $\mathbb{P}(Y(0)_T = i) \propto \mu_i q_i$, so that it has to be equal to ν_i. □

3.2 Coupling from the past for $Y(n)$

The algorithm to sample $Y(n)$ is based on a procedure `simulate`$(E, \mathbf{m})$ whose arguments are E a set of states, and one move $\mathbf{m}$ in $\{\pm\mathbf{e}_1, \ldots, \pm\mathbf{e}_d\}$. `simulate` outputs a set of states F (=`simulate`$(E, \mathbf{m})$) such that $\phi(E, \mathbf{m}) \subset F$.

Once the elementary procedure `simulate` is known, the perfect sampling can be achieved, using coupling from the past. This is done in Algorithm 2. It was shown in [Propp and Wilson, 1996] that if this algorithm (called $PSA(Y)$) terminates, then its output is distributed according to the stationary distribution of $Y(n)$.

Lemma 1 ([Propp and Wilson, 1996])**.** *Under the foregoing assumptions, the output of Algorithm $PSA(Y)$ is distributed according to the stationary distribution of $Y(n)$, if the algorithm terminates.*

A naive implementation of the function `simulate`$(E, \mathbf{m})$ consists in computing the function $\phi(x, \mathbf{m})$ for all $x \in E$. However, its time and space complexity is in $O(|E|)$. Since the size of E is N^d at the start, when $E = \mathcal{S}$, this will be unacceptable, so that compact ways to construct the procedure `simulate` are needed if we want the perfect sampling algorithm to be effective.

3.3 Non-monotonicity and intervals

If there are no forbidden arcs, then the Markov chain Y is monotone in the following sense. Using the previous coupling, $Y_1(n) \preccurlyeq Y_2(n)$ implies $Y_1(n+1) \preccurlyeq Y_2(n+1)$.

The random walk with forbidden arcs is not monotone as shown by the following example. Consider two coupled random walks Y^1 and Y^2, with starting states $Y^1(0) = \mathbf{y}$ and $Y^2(0) = \mathbf{y} - \mathbf{e}_1$ respectively. Consider the case where the arc from $\mathbf{y}$ to $\mathbf{y} + \mathbf{e}_2$ is forbidden, then if the

Algorithm 2: Sampling algorithm of $Y(n)$ $(PSA(Y))$

Data: Function `simulate`
Result: A state F sampled from the stationary distribution of Y

```
1  begin
2      t := 1;
3      generate event m_0;
4      repeat
5          E := S;
6          for i = t − 1 downto 0 do
7              E := simulate(E, m_{−i});
8          t := 2t;
9          generate events m_{−t+1}, ..., m_{−t/2};
10     until coalescence (E is reduced to a unique state);
11     return E;
```

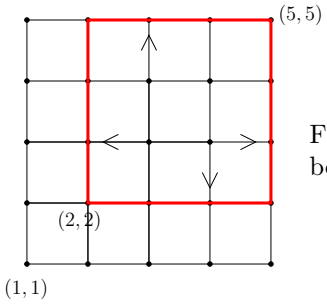


Figure 3: No contraction of upper and lower bounds, in the presence of forbidden arcs

next random move happens to be $\mathbf{e}_2$, then $Y^1(1) = \mathbf{y}$ and $Y^2(1) = \mathbf{y} - \mathbf{e}_1 + \mathbf{e}_2$ respectively. So that $Y^1(0) \preccurlyeq Y^2(0)$ but $Y^1(1)$ and $Y^2(1)$ are not comparable.

When the Markov chain is not monotone, one classical way to sample its stationary distribution is to simulate upper and lower bounds of all the trajectories of the chain instead of its extreme states. Such upper and lower bounds form *intervals* of $\mathcal{S}$.

in [Busic et al., 2008], the authors show how such intervals can be built iteratively considering a non-monotone Markov chain defined by $Y(n+1) = \phi(Y(n), U(n+1))$ over a state space equipped with a complete lattice structure.

Given an interval $E = [\mathbf{a}, \mathbf{b}]$, then the next interval can be defined as

$$\big[\inf_{\mathbf{z}\in[\mathbf{a},\mathbf{b}]} \phi(\mathbf{z},\mathbf{m})\ ,\ \sup_{\mathbf{z}\in[\mathbf{a},\mathbf{b}]} \phi(\mathbf{z},\mathbf{m})\big].$$

This approach does not work here because intervals may never collapse. One such case is presented in Figure 3.

In this case, starting from the extreme points $(1,1)$ and $(5,5)$, and under any sequence of moves, the lower bound will never go over $(2,2)$ and the upper bound will never depart from $(5,5)$.

One way to solve this problem is to replace intervals with more complex data structures that still capture all trajectories but remain compact and coalesce with probability one. Here, the structure we chose is composed of one interval and a set of isolated points.

Suppose the set E is made of one interval $I = [\mathbf{x}_1, \mathbf{x}_2]$ and a set of isolated points $\mathbf{y}_1, ..., \mathbf{y}_k$.

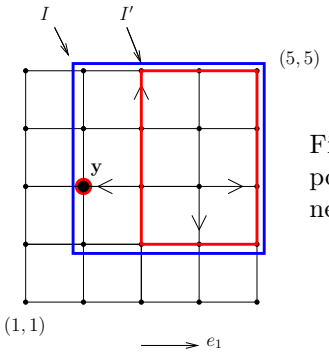


Figure 4: Executing `simulate` over a set E composed of one interval I under move $\mathbf{e}_1$ creates a new interval I' and one isolate point $\mathbf{y}$.

Then, the execution of `simulate`$(E, \mathbf{m})$ transforms E in the following manner (with $\mathbf{m} \in \{\pm\mathbf{e}_1, \ldots, \pm\mathbf{e}_d\}$):

1. The new interval $I' = [\mathbf{x}'_1, \mathbf{x}'_2]$ is such that $\mathbf{x}'_1 = (\mathbf{x}_1 + \mathbf{m}) \wedge \mathbf{N} \vee \mathbf{1}$, $\mathbf{x}'_2 = (\mathbf{x}_2 + \mathbf{m}) \wedge \mathbf{N} \vee \mathbf{1}$ (forbidden arcs are not taken into account).
2. For all the isolated points, ϕ is applied (forbidden arcs are taken into account): $\forall 1 \leqslant j \leqslant k$, $\mathbf{y}'_j = \phi(\mathbf{y}_j, \mathbf{m})$
3. Finally, new isolated points are created: If move $\mathbf{e}_i$ is forbidden in a point $\mathbf{y} \in I - I'$, then $\mathbf{y}$ becomes a new isolated point.

This construction is illustrated in Figure 4, where one isolated point is created by moving the interval to the right.

Theorem 2. *Algorithm 2 terminates with probability one and outputs a sample of the chain $Y(t)$ under its stationary distribution.*

Proof. Using the foregoing notations, Let us consider $E' =$`simulate`$(E, \mathbf{m})$, where $E = I \cup \{\mathbf{y}_1, ..., \mathbf{y}_k\}$. For all isolated points, $\mathbf{y}'_j = \phi(\mathbf{y}_j, \mathbf{m}) \in E'$ by construction. As for the points $\mathbf{x} \in I$, two cases can occur. If the move from $\mathbf{x}$ to $\mathbf{x}+\mathbf{e}_i$ is possible, then $\phi(\mathbf{x}, \mathbf{m}) = (\mathbf{x}+\mathbf{m}) \wedge \mathbf{N} \vee \mathbf{1} \in I'$ by monotonicity. Else, if the move from $\mathbf{x}$ to $\mathbf{x}+\mathbf{e}_i$ is forbidden, then $\phi(\mathbf{x}, \mathbf{m}) = \mathbf{x}$ and if $\mathbf{x} \notin I'$, it becomes a new isolated points. In all cases, $\phi(E, \mathbf{m}) \subset E'$ and from Lemma 1, Algorithm 2 outputs a sample of the chain $Y(t)$ under its stationary distribution whenever it terminates.

To prove termination, let us consider the evolution over time of the set E. It is made of one interval and a set of points. The evolution of the interval does not depend on the forbidden arcs. Under the following sequence of moves,

$$\mathbf{e}_1, \ldots, \mathbf{e}_1, \mathbf{e}_2 \ldots, \mathbf{e}_2, \ldots, \mathbf{e}_d, \ldots, \mathbf{e}_d$$

where each move is repeated N times, the initial interval $I = [\mathbf{1}, \mathbf{N}]$ collapses in point $\mathbf{N}$. Using Borel-Cantelli Lemma, this proves that the interval will collapse with probability one into a single point. From this collapse time on, the set E will be made of a set of isolated points that will coalesce with probability one by irreducibility of the chain (see Appendix A for the details). This part of the proof is illustrated by Figure 5. □

A detailed implementation of procedure `simulate` is given in Algorithm 3. It is based on a dual point of view. The main loop in Algorithm 3 iterates through the list of forbidden arcs instead of iterating through the list of isolated points in E as suggested by the previous description. Using hash tables and a representation of the interval using relative coordinates instead of absolute ones, its time complexity is independent of the number of isolated points and its space complexity is linear in Nk (k is the number of forbidden arcs), see Lemmas 4 and 5.

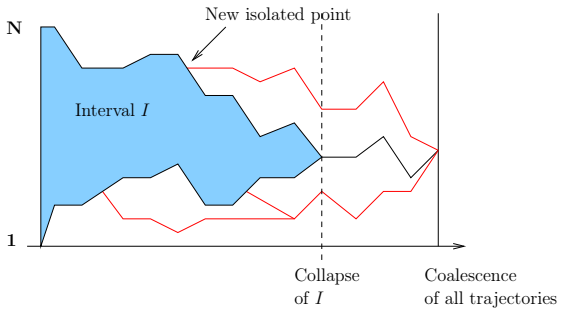


Figure 5: Illustration of the two phases of coalescence using `simulate`. First, new isolated points are created until the interval collapses. After the collapse of the interval, the points merge up to global coalescence.

4 Coalescence and Complexity analysis

We will use the $O(.)$ notation with several variables with the following meaning: $x(N,k,d) = O\left(f(N,k,d)\right)$ means

$$\exists \beta \in \mathbb{R},\ \forall N,k,d \in \mathbb{N},\ d \leq N \Rightarrow x(N,k,d) \leq \beta f(N,k,d).$$

4.1 Coalescence time without forbidden arcs

The goal of this part is to bound the execution time of our algorithms. We first analyze the coalescence time without forbidden arcs (that also corresponds to the collapse time of the interval in E), then the complexity of the related part of the algorithms before giving both space and time complexity in the general case.

We will call C the coalescence time in a grid of dimension 1. This is the number of iterations of the function ϕ needed for all the Markov chains starting from all the points in $[1,N]$ to coalesce in a single point.

Although the following result concerns a simple random walk on the line, we have not been able to find a similar asymptotic formula in the literature. The best bound we could find is based on hitting times and involves the spectral gap. It is of the order $O\left(1-\frac{1}{N^3}\right)^T$ (see [Aldous and Fill, 2002]). This is not good enough to assert the complexity of our algorithms, given below.

Lemma 2 (Coalescence in dimension 1)**.** *For any $T>0$, $\mathbb{P}(C>T) \leqslant \cos^T\left(\frac{\pi}{N+1}\right)\left(1+O(\frac{1}{N^2})\right)$.*

Proof. The proof is technical and is based on the spectral decomposition of the transition matrix. It is given in appendix B. □

The case of the random walk in dimension d follows. let C_d be the coalescence time in dimension d. The expectation of C_d can be estimated, based on Lemma 2. This also provides the time complexity of Algorithm 2.

Lemma 3 (Coalescence in dimension d)**.** *Let us consider a random walk in a grid with no forbidden arcs. Let C_d be its coalescence time.*
(i) The sampling time τ_Y, of Algorithm $PSA(Y)$ (number of calls to `simulate`*) is smaller than $4C_d$.*
(ii) The expected coalescence time satisfies $\mathbb{E}[C_d] = O\left(N^2 d \log d\right)$.

Proof. (i) The fact that $\tau_Y \leqslant 4C_d$ follows from the doubling period trick used in Algorithm 2. Indeed, $\tau_Y = 1+2+\cdots+2^c$ where 2^c is the smallest power of 2 larger than C_d. This implies that $\tau_Y \leqslant 4C_d$.
(ii) The order of magnitude of $\mathbb{E}[C_d]$ can be computed as follows. The coalescence in dimension

d exactly occurs when coalescence occurs in each dimension, each of them being independent, and chosen with probability $1/d$ at each step. Therefore, $\mathbb{E}[C_d] = d\mathbb{E}[\max(C^{(1)}, \ldots, C^{(d)})]$ where $C^{(i)}$ denotes the coalescence time in the i-th dimension. By independence,

$$\begin{aligned}\mathbb{P}(\max(C^{(1)}, \ldots, C^{(d)}) > T) &= 1 - \mathbb{P}(\max(C^{(1)}, \ldots, C^{(d)}) \leqslant T) \\ &= 1 - (1 - \mathbb{P}(C^{(1)} > T))^d,\end{aligned}$$

$$\mathbb{E}[C_d] = d\sum_{T=0}^{\infty} \mathbb{P}(\max(C^{(1)}, \ldots, C^{(d)}) > T) = d\sum_{T=0}^{\infty} \Big(1 - (1 - \mathbb{P}(C^{(1)} > T))^d\Big).$$

Using Lemma 2, $\mathbb{E}[C_d] \leqslant d\sum_{T=0}^{\infty}(1 - (1 - h^T)^d) + \varepsilon$, where $h = \cos(\frac{\pi}{N+1})$ and ε is negligible with respect to the first term. From this point, one can show (see appendix C)

$$\mathbb{E}[C_d] \leqslant d + \frac{d\log d}{-\log h} + \varepsilon,$$

using the fact that $\log h \approx \frac{-\pi^2}{N^2}$, we get $\mathbb{E}[C_d] = O\left(N^2 d \log d\right)$. □

4.2 Number of rejections, without forbidden arcs

Let us now focus on the sampling algorithm for X, $PSA(X)$. The reject probability is the probability that the next move cannot be taken by the walker who is in a stationary state (of Y). In general this probability is bounded by $\frac{1}{2d}$ (for example if the walker has all moves forbidden but one). So that the expected number of rejections is always bounded by $2d$. However this bound is very loose. By using the fact that the stationary measure of Y is uniform over all states, then the probability p_R that the next move is blocked can be computed by considering each dimension independently:

$$p_R = \sum_{i=1}^{2d} \frac{1}{2d}\mathbb{P}(\text{move blocked} \mid \text{move } i \text{ chosen }) = \sum_{i=1}^{2d} \frac{1}{2d}\frac{1}{N} = \frac{1}{N}.$$

Therefore, the number of runs R up to the first non-rejection is Bernoulli distributed and $\mathbb{E}[R] = \frac{N}{N-1}$.

This allows us to state our main result in the case where no arc is forbidden. Actually, as numerical experiments suggest, we believe this complexity is true as long as the random walk has a large connectivity degree.

Theorem 3. *With no forbidden arcs, the expected sampling time $\mathbb{E}[\tau_X]$ for Algorithm $PSA(X)$ (Algorithm 1) is such that $\mathbb{E}[\tau_X] = O(N^2 d \log d)$.*

Proof. The sampling time for Algorithm $PSA(X)$ is $\tau_X = \sum_{i=1}^{R} \tau_Y(i)$, where $\tau_Y(i)$ is the sampling time of the i-th run of Algorithm $PSA(Y)$. It satisfies the following assumptions: The sampling times $\tau_Y(i)$ form an iid sequence of positive random variables with finite expectations, and the number of runs, R, is a stopping time of the filtration generated by the sampling algorithm. Using Wald's equation, the expected sampling time is $\mathbb{E}[\tau_X] = \mathbb{E}[R]\ \mathbb{E}[\tau_Y] = O(N^2 d \log d)$, by Lemma 3. □

4.3 Complexity with forbidden arcs

For the general case, the coalescence time is harder to estimate analytically because it does not only depend on the number of forbidden arcs but also on their actual configuration. For example, using only $2d$ forbidden arcs, it is possible to cut the state space into several disconnected components, in which case the coalescence time of the algorithm will be infinite.

Even if it is not possible to give tight bounds for the coalescence time of Algorithm 2, one can still use Lemma 3 to bound the memory space used by the algorithm. Also note that the time complexity (number of calls to `simulate` up to coalescence of the interval is the same as without forbidden arcs.

Let us call $|E|$ the number of isolated points composing the sets E used by $\texttt{simulate}(E, \cdot)$ in Algorithm 2. This size can be bounded as a function of the number k of forbidden arcs.

Lemma 4. *If forbidden arcs are chosen randomly, uniformly among all arcs in the grid and if k is the expected number of forbidden arcs, then the maximal size $|E|$ of the set E, is bounded in expectation: $\mathbb{E}[\,|E|\,] \leq \frac{kN}{\pi^2} + O(kd + \frac{kN}{d})$.*

Proof. The proof is based of a combinatorial argument to bound the number of isolated points created at each execution of `simulate`. It is reported in Appendix D. □

The inner loop in Algorithms 1 and 2 uses the procedure `simulate` that is therefore the main component in the time and space complexity. A detailed version of the procedure `simulate` is given in Algorithm 3.

The main idea is to replace the iteration through all isolated points (see Sec. 3.3, item 2) by an iteration through all the forbidden arcs (line 2 in Algorithm 3).

We use two hash tables $\mathcal{H}$ and $\mathcal{A}$ to store, access and merge isolated points in constant time: $\mathcal{H}$ for the forbidden arcs and $\mathcal{A}$ for each border. They are treated as sets in the algorithm.

Finally, we use relative coordinates so that we do not have to access every element at each move, only those encountering the forbidden arcs. More precisely, we have a reference point $\mathbf{r}$ moving freely on $\mathbb{Z}^d$ (in fact on $[-N, 2N]^d$), and all the other points are coded by their coordinates relatively to it (in $[-N, N]^d$). This allows us to virtually move the isolated points as blocks, without recalculating every hash of coordinates.

In the details, the loop in line 2 is the search of obstacles (*i.e.* forbidden moves and border) and test if they block a isolated point (lines 3 and 4) or if they separate an element from the interval (lines 6 and 7). Line 6 implies that $\mathbf{k}$ *exits* $[\mathbf{a}, \mathbf{b}]$, more precisely, the head of the forbidden arc is in the interval before the move and no longer after.

The loop in line 8 concerns the isolated points that hit the border. Because of the relative notation, we can obtain them by checking the set of indexes $(i, N - r_i)$ containing every element with a relative i^{th} coordinate of $N - r[i]$, hence an absolute one equal to N, at the border. We do the same for the interval.

In line 13 the reference point is moved along $\mathbf{m}$. The last loop (lines 14 to 17) consists in the insertion of each extracted point in the hash tables after having compensated the global move, if it has not merged with the interval.

Lemma 5. *Let k be the number of forbidden arcs in the grid, each execution of $\texttt{simulate}(E, \mathbf{m})$ takes time $O(d+k)$ and is independent of the number of isolated points in E. Its space complexity is $O(dN + \mathbb{E}[\,|E|\,])$, hence $O(dN + kN)$ in the random case.*

Proof. The extraction (line 2) is done by a search of the forbidden arcs and the border aligned against the move. The former contains an average (depending on the direction chosen) of $\frac{k}{2d}$ forbidden arcs. It will make $(1 + \frac{k}{d})$ tests and add at most $1 + \frac{k}{d}$ elements to the waiting list.

Algorithm 3: procedure `simulate` (detailed)

Data:
1. Forbidden arcs, coded by their head point sorted in $2d$ sets $\mathcal{K}_{0\leq i<2d}$ by their direction.
2. An initially empty waiting list $\mathcal{W}$
3. The move $\mathbf{m} = +\mathbf{e}_i$ *(the case* $\mathbf{m} = -\mathbf{e}_i$ *is obtained by replacing* N *by* 1 *and* $\mathbf{b}$ *by* $\mathbf{a}$ *in lines 1-17)*
4. The interval, $I = [\mathbf{a}, \mathbf{b}]$ (initialized at $t = 1$ to $\mathbf{a} = \mathbf{1}$ and $\mathbf{b} = \mathbf{N}$)
5. A reference point $\mathbf{r}$ (initialized at $t = 1$ to $\mathbf{r} = \mathbf{1}$)
6. A hash table $\mathcal{H}$ coding the relative positions of the isolated points
7. Arrays $\mathcal{A}_{1\leqslant i\leqslant d}$, storing the relative positions of the isolated points on the border in each dimension, modulo N.

```
1  begin
2      foreach k ∈ K_m do
3          if k ∈ H(k − r) then
4              remove k from H and from A, and add to W      //blocked isolated
5                                                            //points
6          if k_i = a_i then
7              insert k in W     //creation of a new isolated point if k exits [a, b]
8      foreach s ∈ A_i[N − r_i] do
9          remove s from H and from A, and add to W    //isolated points blocked
10                                                     //by the border
11     if b_i = (N − r_i) then
12         b ← b − m                          //compensate for the update of r
13     r ← r + m
14     foreach x ∈ W do
15         x ← x − m
16         if x_i ≠ b_i then
17             insert x in H and A       //only keep isolated points outside interval
```

Each test can be done in time $O(d)$, and hash tables access times in $O(d)$. Hence the time taken by this step is in $O(d + k)$.

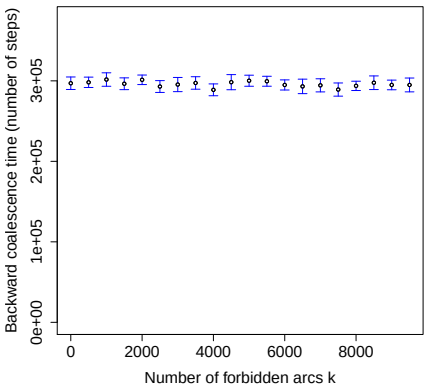
Updates and reinsertions are similar to the first step : at most $O(1 + \frac{k}{d})$ elements with operations costing $O(d)$ in time. The third step hence takes a time $O(|\mathcal{W}|) = O(d + k)$. Overall, the global time cost is $O(d + k)$.

As for the space complexity, one hash table with at most $|E|$ element, and d arrays of size N need a space of $O(|E| + d\ N)$ in memory. The only other variable with a non fixed size is the waiting list; its size is bounded by $\max \mathcal{K}_d + 1 < k + 1$, which is below $|E|$. □

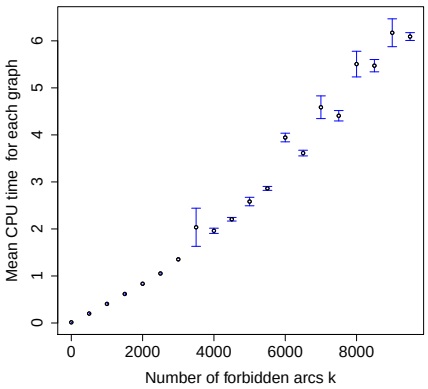
5 Numerical experiments

We have implemented Algorithm 1 in C++ (700 lines of code) to validate the approach and test the actual sampling time under several values of the parameters N (the span of the state space), d the dimension of the state space) and k (the number of forbidden arcs).

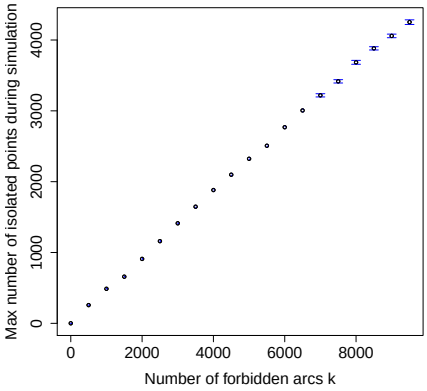
To assert the performance of the algorithm with forbidden arcs, for each fixed value of k



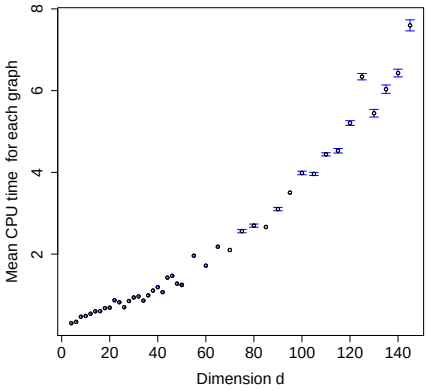
(a) Coupling time independent of number of forbidden arcs, k.



(b) CPU time for one sample grows linearly with k.



(c) Number of isolated points grows linearly with k.



(d) CPU time for one sample is a convex function of d.

Figure 6: Experimental evaluation of Algorithm $PSA(X)$.

(the number of forbidden arcs), we have generated 20 random grids, choosing the forbidden arcs uniformly. Then, we have run PSA(X) 50 times over each grid. The 95 % confidence intervals are reported in all the following figures. They all remain very small, confirming a small variance in the performance of the algorithm. Finally, in most experiments, the state space is greater than 10^{20} (very big number of states that cannot be handled by computers). However each sample has been obtained within a few seconds (with $N \geqslant 100$, $d \geqslant 10$) over a desktop PC (2.3 GHz Intel Core I7, with 16 Go of memory).

We have shown in Sec. 4.1 that the coalescence time of Algorithm 1 is in $\mathcal{O}(N^2 d \log d)$ in the absence of forbidden arcs. We show in Fig. 6.(a) that the presence of forbidden arcs does not increase coalescence time, which suggests that Theorem 3 generalizes to (strongly connected) random graphs with forbidden arcs. This observation empirically justifies the computation of the coalescence time without forbidden arcs as one measure of complexity in Sec. 4. This is further corroborated by another measurement: the number of isolated points *at the collapse time* of the interval averaged 1.2, and very scarcely exceeded 2. This means that when the interval collapses, all the isolated points that have been created have also merged in most cases. The second phase of coalescence is therefore very short. It is null when all the points have already merged (majority of the cases), or rather short when a few points remain.

However, the coalescence time is not the only factor that affects the complexity of Algorithm 1. We have executed the program over a standard PC. Figure 6.(b) shows that the actual sampling time (CPU time, measured with `ctime` library). It grows almost linearly with the number k of forbidden arcs, as predicted by our complexity analysis. Indeed, as detailed in Lemma 5, the procedure `simulate` iterates over a set K_m of forbidden arcs in a given direction m (line 2 of Algorithm 3). On average there are $\frac{k}{2d}$ forbidden arcs in each direction, which explains the linear complexity in k. The super-linearity trend for small values of k may come from hidden negligible factors in the complexity.

Figure 6.(c) reports the number of isolated points ever created during the execution as a function of k, the number of forbidden arcs. As predicted by Lemma 4, this number also grows linearly with k, and has a very small variance over the executions.

Finally, Figure 6.(d) reports the execution time (in seconds) to obtain one stationary sample over a grid of size 100^d, with 1000 forbidden arcs, as a function of d. This performance metric is important with application to interacting particles in mind, because $d/3$ represents the number of particles in the 3-dimension space that are simulated. Our complexity analysis gives an asymptotic complexity bounded by $O(d^2 \log d)$ while the measures seems to have a super-linear but sub-quadratic behavior.

6 Improvements and Generalizations

6.1 Domination

Algorithm 1 can be further improved when the forbidden arcs have special properties.

In the case where all forbidden arcs are of the type $+\mathbf{e}_1, \ldots, +\mathbf{e}_d$ (in other words, no moves from $\mathbf{x}$ to $\mathbf{x}-\mathbf{e}_i$ is forbidden), then the continuous time chain $Y(t)$ is dominated by a new random walk $D(t)$ over the same state space, but where so moves are forbidden.

Here domination means that there exists a coupling of the chains such that if $Y(0) \preceq D(0)$, then $Y(t) \preceq D(t)$, for all t.

This coupling is easy here: The function β constructing D is equal to ϕ in all states $\mathbf{x}$ and for all valid moves $\pm\mathbf{e}_1, \ldots, \pm\mathbf{e}_d$: $\phi(\mathbf{x}, \mathbf{e}) = \beta(\mathbf{x}, \mathbf{e})$. The only difference is when a move (say $\mathbf{e}$) is forbidden for Y, then $\phi(\mathbf{x}, e) = \mathbf{x}$ but $\beta(\mathbf{x}, \mathbf{e}) = \mathbf{x} + \mathbf{e}$.

This allows us to further improve the sampling algorithm by using the same technique as in [Busic et al., 2012]. In that case one can start the simulation by starting from points $(1, ..., 1)$ and a random point, uniform over all the state space instead of $(1, ..., 1)$ and $(N, N, ..., N)$.

Lemma 6. *Without forbidden arcs, the coupling time τ_{Dom} in the domination algorithm 4 is smaller than the coupling time without domination τ_X. It satisfies $\mathbb{E}[\tau_{Dom}] = \gamma\mathbb{E}[\tau_X]$, where $2/3 \leqslant \gamma < 1$, asymptotically.*

Proof. The fact that when starting from an arbitrary point $\mathbf{m} \preccurlyeq \mathbf{N}$ and $\mathbf{1}$ one gets a smaller coalescence time follows by monotonicity of the chain when no arc is forbidden. In dimension 1 and using the definition of function $F(N,T)$, the coupling time $C_1^{Dom}(M)$ of two walks, starting respectively in states 0 and M under domination satisfies $\mathbb{P}(C_1^{Dom}(M) > T) = F(N,T) - F(N-1,T) - F(M,T) + F(M-1,T)$. Therefore, by deconditioning on the initial starting point M, we get

$$\begin{aligned}\mathbb{E}[C_1^{Dom}] &= \sum_{M=1}^{N} \frac{1}{N} \sum_{T} (F(N,T) - F(N-1,T) - F(M,T) + F(M-1,T)) \\ &= \frac{2}{3}\frac{\pi^2}{(N+1)^2} + O(\frac{1}{N^4}).\end{aligned}$$

In dimension $d > 1$, by conditioning on the starting point $(M_1, \ldots, M_d)$, one gets for the expected coupling time, denoted $\mathbb{E}[C_d^{Dom}(M_1, \ldots, M_d)]$,

$$\mathbb{E}[C_d^{Dom}(M_1, \ldots, M_d)] = d\mathbb{E}[\max(C_1^{Dom}(M_1), \ldots, C_1^{Dom}(M_d)]$$

Using Jensen inequality, and by deconditioning,

$$\begin{aligned}\mathbb{E}[C_d^{Dom}] &\geqslant d\mathbb{E}[\max(C_{1_1}^{Dom}, \ldots, C_{1_d}^{Dom})] \\ &\geqslant \frac{2}{3} d\mathbb{E}[\max(C_{1_1}, \ldots, C_{1_d})].\end{aligned}$$

where $C_{1_1}^{Dom}, \ldots C_{1_d}^{Dom}$ and $C_{1_1}, \ldots C_{1_d}$ are independent versions of the coupling time in one dimension under domination and without domination, respectively.

□

6.2 Blocks of forbidden arcs

The complexity of the algorithm depends of the number of forbidden arcs. In some cases, forbidden arcs may be described in a compact way as *clusters* of forbidden arcs (for example all arcs in one hyperplane, but one). In this case, the solution is to treat them the same way we did for the borders: dedicate the specific data structures (like $\mathcal{K}_\rangle$ and A_i) to each forbidden cluster, define the inclusion test and add them to the loop in procedure `simulate` (such as lines 8-9 of Algorithm 3).

This will only increase the time cost as if we had added one forbidden arc so the memory cost will become: $O(kN(d+f))$ with f the number of clusters.

Algorithm 4: Sampling algorithm with domination

Data: Functions `simulate` and β.
Result: A state E sampled from the stationary distribution of X

begin
  repeat
    Sample $\mathbf{z}$, uniformly distributed in $\mathcal{S}$;
    $t := 1$; generate $\mathbf{m}_0, \mathbf{m}_1$ uniformly in $\mathcal{E}$;
    repeat
      for $k = \lfloor t/2 \rfloor$ **to** $t - 1$ **do**
        $\mathbf{z} := \beta(\mathbf{z}, \mathbf{m}_{-k})$;
        $\mathbf{v}_{-k} := -\mathbf{m}_{-k}$ (to couple forward and backward trajectories);
      $E := [\mathbf{1}, \mathbf{N}] \cup \{\emptyset\}$;
      for $i = t - 1$ **downto** 0 **do**
        $E :=$`simulate`$(E, \mathbf{v}_{-i})$;
      $t := 2t$;
      generate $\mathbf{m}_{-t+1}, \ldots, \mathbf{m}_{t/2}$, uniformly in $\mathcal{E}$.
    until *E is a isolated point*;
  until $Y^{(1)} \neq \phi(Y^{(1)}, \mathbf{e}_1)$;
  return *F;*

References

[Aldous and Fill, 2002] Aldous, D. and Fill, J. (2002). *Reversible Markov Chains and Random Walks on Graphs*. University of California, Berkeley.

[Brémaud, 1998] Brémaud, P. (1998). *Markov chains Gibbs fields, Monte Carlo simulation, and queues*. Springer.

[Broadbent and Hammersley, 1957] Broadbent, S. and Hammersley, J. (1957). Percolation processes i. crystals and mazes. *Proceedings of the Cambridge Philosophical Society*, 53:629–641.

[Busic et al., 2012] Busic, A., Gaujal, B., and Perronnin, F. (2012). Perfect Sampling of Networks with Finite and Infinite Capacity Queues. In Al-Begain, K., Fiems, D., and Vincent, J.-M., editors, *19th International Conference on Analytical and Stochastic Modelling Techniques and Applications (ASMTA) 2012*, volume 7314 of *Lecture Notes in Computer Science*, pages 136–149, Grenoble, France. Springer.

[Busic et al., 2008] Busic, A., Gaujal, B., and Vincent, J.-M. (2008). Perfect simulation and non-monotone Markovian systems. In *3rd International Conference Valuetools'08*, Athens, Greece. ICST.

[Kendall and Møller, 2000] Kendall, W. S. and Møller, J. (2000). Perfect simulation using dominating processes on ordered spaces, with application to locally stable point processes. *Advances in Applied Probability*, 32(3):844–865.

[Levin et al., 2008] Levin, D., Peres, Y., and Wilmer, E. (2008). *Markov Chains and Mixing Times*. AMS, American Mathematical Society.

[Plateau and Stewart, 1997] Plateau, B. and Stewart, W. J. (1997). Stochastic automata networks. In *Computational Probability*, pages 113–152. Kluwer Academic Press.

[Propp and Wilson, 1996] Propp, J. G. and Wilson, D. B. (1996). Exact sampling with coupled Markov chains and applications to statistical mechanics. *Rand. Struct. Alg.*, 9(1-2):223–252.

[Schwartz et al., 2002] Schwartz, N., Cohen, R., ben Avraham, D., Barabasi, A.-L., and Havlin, S. (2002). Percolation in directed scale-free networks. *PHYSICAL REVIEW E*, 66.

A Appendix: Proof of Theorem 2

In the proof of Theorem 2, the only remaining point is the fact that for any coupled trajectories $Y(n+1) = \phi(Y(n), \mathbf{m}_n)$ and $Z(n+1) = \phi(Z(n), \mathbf{m}_n)$ coalesce in finite time for any starting positions $Y(0) = \mathbf{a}_0$ and $Z(0) = \mathbf{b}_0$, with probability one.

To do so, we construct a specific sequence of moves, s, such that $Y(n)$ and $Z(n)$ coalesce under s, and then we use the Borel-Cantelli Lemma to assert that coalescence will happen with probability one. This sequence s of moves is constructed step by step as follows.

First, let us define the pseudo-distance $d(\mathbf{a}, \mathbf{b})$ between two points $\mathbf{a}$ and $\mathbf{b}$ in $\mathcal{S}$ as the smallest number of moves to go from $\mathbf{a}$ to $\mathbf{b}$. This distance is always finite by the irreducibility assumption. Let us also define $s(\mathbf{a}, \mathbf{b})$ to be a shortest sequence of moves to go from $\mathbf{a}$ to $\mathbf{b}$ (by definition $s(\mathbf{a}, \mathbf{b})$ is made of $d(\mathbf{a}, \mathbf{b})$ moves).

Let us now start with two initial points $\mathbf{a}_0$ and $\mathbf{b}_0$, and let us choose $s_0 = s(\mathbf{a}_0, \mathbf{b}_0)$. After this sequence of moves, $\mathbf{a}_0$ goes to $\mathbf{a}_1 = \mathbf{b}_0$ and $\mathbf{b}_0$ goes to a new point $\mathbf{b}_1$. At this point, two cases can occur: either the trajectory starting from $\mathbf{b}_0$ encountered at least one forbidden arcs, or it did not.

1. In the first case, $d(\mathbf{a}_1, \mathbf{b}_1) < d(\mathbf{a}_0, \mathbf{b}_0)$, because the path from $\mathbf{b}_0$ to $\mathbf{b}_1$ contains less than $d(\mathbf{a}_1, \mathbf{b}_1)$ moves, some of them having been forbidden.

2. In the second case, we only have $d(\mathbf{a}_1, \mathbf{b}_1) \leqslant d(\mathbf{a}_0, \mathbf{b}_0)$ but the vector $\overrightarrow{v_0} = \mathbf{b}_0 - \mathbf{a}_0$ is equal to $\overrightarrow{v_1} = \mathbf{b}_1 - \mathbf{a}_1$.

Next, we show that by repeating this construction using sequences of move

$$s = s(\mathbf{a}_0, \mathbf{b}_0), s(\mathbf{a}_1, \mathbf{b}_1), s(\mathbf{a}_2, \mathbf{b}_2), \ldots,$$

the distance $d(\mathbf{a}_i, \mathbf{b}_i)$ will eventually decrease, or equivalently, that the second case can only happen a finite number of times. Indeed, if the second case happens n times consecutively after m steps, we get $\mathbf{a}_{m+n} = \mathbf{a}_m + n\overrightarrow{v_m}$, that eventually gets out of the grid if n is too large. Therefore the distance can only remain constant during a finite number of iterations.

Since the distance $d(\mathbf{a}_i, \mathbf{b}_i)$ decreases under the above sequence of moves $s(\mathbf{a}_1, \mathbf{b}_1), s(\mathbf{a}_2, \mathbf{b}_2), \ldots$, it will eventually reach 0, meaning coalescence between $\mathbf{a}_i$ and $\mathbf{b}_i$ in finite time.

B Proof of Lemma 2

Proof. Let us consider a linear grid of size N and dimension $d = 1$. We will compute a bound on the cumulative probability of coalescence, *i.e.* the function that returns the probability that coalescence did not occur within their first T moves.

We will start by changing the point of view. Let us consider a random walk $X(n)$ on $\mathbb{Z}$ (with no boundary). At time t, we note c_t and C_t the smallest and biggest values respectively, covered by X since the beginning (we call the difference $C_t - c_t$ the *amplitude* of X at time t). In parallel, let us also consider an interval $I(t) = [a_t, b_t]$ on $[1, N]$ with the same moves (coupled to X). At time 0, $I(0) = [1, N]$. Using the coupling, one can show that $X(t) = C_t \Leftrightarrow b(t) = N$ and $X(t) = c_t \Leftrightarrow a(t) = 0$. From this, the amplitude of $X(t)$ increases if and only if the size of the interval $I(t)$ decreases.
Hence, by induction, at any time t, N minus the size of the interval $I(t)$ corresponds to the amplitude reached by $X(t)$.

To compute the coalescence time we consider now the probability that the amplitude of the trajectory of a point is below N during a random walk of duration T.

We count the paths of length T, starting at 0, for which there exists an interval of size N containing it (meaning its amplitude is below N).

For each path $\mathcal{P}$ of amplitude less than N, we can list all the N-sized and $(N-1)$-sized intervals containing it. We pair these intervals by their left extremity. Exactly one interval of size N remains alone (the one with the left most extremity containing $\mathcal{P}$). If we denote by $F(N,T)$ the sum for every path of the number of intervals containing it (i.e. the number of couples of the form *(possible path, N-sized interval containing it)*, divided by 2^T (the probability of each path), the number of paths whose amplitude has a size smaller than N is $2^T\,(F(N,T) - F(N-1,T))$.

Using the transformation introduced in the first paragraph, this implies

$$P(C > T) = F(N,T) - F(N-1,T). \tag{1}$$

By taking as origin the left extremity of the intervals instead of the starting point (a simple change of variable), we can see that $2^T F(N,T)$ is the number of paths included in a (fixed) interval of size N.

This can be computed as the sum of all coefficients of the power matrix $A(N)^T$ where $A(N)$ is the square matrix of size N with $A(N)_{ij} = \delta_{|i-j|=1}$:

$$A(N) = \begin{pmatrix} 0 & 1 & \cdots & 0 \\ 1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 \\ 0 & \cdots & 1 & 0 \end{pmatrix}.$$

To compute the T-th power for $A(N)$, we can use the fact that $A(N)$ is diagonalizable: Let P be such that $A(N) = PDP^{-1}$ with D diagonal.

$$\begin{aligned} 2^T F(N,T) &= \sum_{1\leq i,j\leqslant N} (A(N)^T)_{ij} = \sum_{1\leq i,j\leqslant N} (PD^TP^{-1})_{ij} \\ &= \sum_{k=1}^{N} \left((\sum_{1\leq i\leqslant N} P_{ik})(\sum_{1\leq j\leqslant N} P^{-1}_{kj}) D^T_k \right) \\ &= \sum_{k=1}^{N} \left(H_k D^T_k \right) \end{aligned}$$

where $H_k \stackrel{\text{def}}{=} (\sum_{1\leq i\leqslant N} P_{ik})(\sum_{1\leq j\leqslant N} P^{-1}_{kj})$.

This matrix $A(N)$ and its eigenvalues are known: they are exactly twice the roots of the N^{th} Chebychev second order polynomial: $D_k = 2\cos(\frac{k\pi}{N+1})$ with the eigenvector $\mathbf{v_k}_i = U_i(D_k)$ (where U_i is the i^{th} Chebychev second order polynomial). Therefore, we get $P_{ij} = \frac{2\sin(\frac{ij\pi}{N+1})}{\sqrt{N+1}}$.

One can notice that the scalar product of two different column vectors of P is null and the norm of these vectors is 1. Hence P has determinant 1 as well. Let $(\mathbf{v_k})_k$ be the the corresponding orthonormal base.

Let $Q(k)$ be the matrix obtained by substituting the k^{th} column by the vector $\mathbb{1}$ in P. One can recognize in $(\sum (P^{-1})_{kj}) = \sum ({}^{\text{t}}\text{com}P)_{kj}$ the development of the determinant of $Q(k)$ by its k^{th} column. Rewriting the vector $\mathbf{1}$ in the base $(\mathbf{v_k})_k$, and using the linearity and anti-symmetry properties of the determinant, We get

$$H_k = \left(\sum_{1\leq i\leq N} P_{ik} \right) \mathbf{v_k}.\mathbf{1} = \left(\sum_{i=1}^{N} P_{ik} \right) \left(\sum_{j=1}^{N} P_{kj} \right) = \frac{2(1-(-1)^k)}{N+1} \frac{\cos^2(\frac{k\pi}{2(N+1)})}{\sin^2(\frac{k\pi}{2(N+1)})}$$

Therefore,

$$\begin{aligned} F(N,T) &= \tfrac{2}{N+1}\sum_{k=0}^{\lfloor\frac{N}{2}\rfloor}\cos^T\left(\tfrac{(2k+1)\pi}{N+1}\right)\cot^2\left(\tfrac{(2k+1)\pi}{2(N+1)}\right) \\ &= \tfrac{2\cos^T\left(\frac{\pi}{N+1}\right)}{N+1}\sum_{k=1}^{\lfloor\frac{N}{2}\rfloor}\cot^2\left(\tfrac{(2k+1)\pi}{2(N+1)}\right) \\ &\quad -\tfrac{2}{N+1}\sum_{k=1}^{\lfloor\frac{N}{2}\rfloor}\cot^2\left(\tfrac{(2k+1)\pi}{2(N+1)}\right)\left(\cos^T\left(\tfrac{\pi}{N+1}\right)-\cos^T\left(\tfrac{(2k+1)\pi}{N+1}\right)\right). \end{aligned}$$

Using the formula

$$\sum_{1\le k\le N, k \text{ odd}}\cot^2\left(\frac{k\pi}{2(N+1)}\right)=\frac{N(N+1)}{2},$$

we get

$$F(N,T)=N(1-\alpha)^T-h(N,T)$$

with

$$h(N,T)=\frac{2}{N+1}\sum_{k=1}^{\lfloor\frac{N}{2}\rfloor}\cot^2\left(\frac{(2k+1)\pi}{N+1}2\right)\left(\cos^T\left(\frac{\pi}{N+1}\right)-\cos^T\left(\frac{(2k+1)\pi}{N+1}\right)\right),$$

and $\alpha \stackrel{\text{def}}{=} 1-\cos\left(\frac{\pi}{N+1}\right)=\frac{\pi^2}{(N+1)^2}+O(N^{-4})$. It can be shown (by calculus and differentiation) that h is a positive function, and for any T, $h(.,T)$ is increasing.

Introducing $\beta \stackrel{\text{def}}{=} 1-\cos(\frac{\pi}{N})\simeq\frac{\pi^2}{N^2}$ and $\gamma \stackrel{\text{def}}{=} \frac{1}{T-1}\left(\sum_{i=0}^{T-1}\alpha^i\beta^{T-i-1}\right)^{\frac{1}{T-1}}$, we get

$$\begin{aligned} F(N,T)-F(N-1,T) &= N(1-\alpha)^T-(N-1)(1-\beta)^T+h(N-1,t)-h(N,t) \\ &\le (1-\alpha)^T+(N-1)((1-\alpha)^T-(1-\beta)^T) \\ &= (1-\alpha)^T+(N-1)(\beta-\alpha)(1-\gamma)^{T-1} \\ &= (1-\alpha)^T+\tfrac{\pi^2(2N-1)}{N(N-1)^2}(1-\gamma)^{T-1} \end{aligned}$$

Taking into account the fact that $\alpha<\beta$ gives $\alpha<\gamma$ and thus $(1-\gamma)^T<(1-\alpha)^T$ for all T. Therefore, $P(C>T)\le(1-\alpha)^T(1+O(\frac{1}{N^2})$. □

C Proof of Lemma 3

The coalescence in dimension d exactly occurs when coalescence occurs in each dimension, each of them being independent, and chosen with probability $1/d$ at each step. Therefore, $\mathbb{E}[C_d]=d\mathbb{E}[\max(C^{(1)},\dots,C^{(d)})]$ where $C^{(i)}$ denotes the coalescence time in the i-th dimension.

We can compute :

$$\begin{aligned} \mathbb{P}(\max(C^{(1)},\dots,C^{(d)})>T) &= 1-\mathbb{P}(\max(C^{(1)},\dots,C^{(d)})\leqslant T) \\ &= 1-\mathbb{P}(C^{(1)}\leqslant T)^d \\ &= 1-(1-\mathbb{P}(C^{(1)}>T))^d. \end{aligned}$$

By definition,

$$\begin{aligned}\mathbb{E}[C_d] &= d\sum_{T=0}^{\infty}\mathbb{P}(\max(C^{(1)},\dots,C^{(d)})>T)\\ &= d\sum_{T=0}^{\infty}\Big(1-(1-\mathbb{P}(C^{(1)}>T))^d\Big).\end{aligned}$$

Using Lemma 2,

$$\mathbb{E}[C_d] = d\sum_{T=0}^{\infty}\Big(1-(1-(1-\alpha)^T)^d\Big)+\varepsilon,$$

where ε is negligible with respect to the first term. We focus on the first term.

$$\begin{aligned}d\sum_{T=0}^{\infty}\big(1-(1-(1-\alpha)^T)^d\big) &\leq d + d\int_{t=0}^{\infty}1-(1-(1-\alpha)^t)^d dt\\ &\leq d + d\int_{t=0}^{\infty}1-\sum_{i=0}^{d}\binom{d}{i}(-1)^i(1-\alpha)^{ti}dt\\ &\leq d + d\int_{t=0}^{\infty}\sum_{i=1}^{d}\binom{d}{i}(-1)^{i-1}(1-\alpha)^{ti}dt\\ &\leq d + d\sum_{i=1}^{d}\binom{d}{i}(-1)^{i-1}\int_{t=0}^{\infty}(1-\alpha)^{ti}dt\\ &\leq d + d\sum_{i=1}^{d}\binom{d}{i}\frac{(-1)^{i-1}}{i}\frac{1}{-\log(1-\alpha)}\\ &\leq d + \frac{d}{-\log(1-\alpha)}\sum_{i=1}^{d}\binom{d}{i}\frac{(-1)^{i-1}}{i}\end{aligned}$$

Let us denote $S_d = \sum_{i=1}^{d}\binom{d}{i}\frac{(-1)^{i-1}}{i}$. By extracting d, $S_d = d\sum_{i=0}^{d-1}\binom{d-1}{i}(-1)^i\frac{1}{(i+1)^2}$.

To compute this sum , let us introduce

$$G_d(x) = \sum_{i=0}^{d-1}\binom{d-1}{i}(-1)^i\frac{x^{i+1}}{(i+1)^2}.$$

Differentiating w.r.t. x yields

$$\begin{aligned}G'_d(x) &= \sum_{i=0}^{d-1}\binom{d-1}{i}(-1)^i\frac{x^i}{(i+1)} = \frac{1}{x}\sum_{i=0}^{d-1}\binom{d-1}{i}(-1)^i\frac{x^{i+1}}{(i+1)}\\ &= \frac{1}{x}\int_0^x\sum_{i=0}^{d-1}\binom{d-1}{i}(-1)^i u^i du = \frac{1}{x}\int_0^x(1-u)^{d-1}du\\ &= \frac{1-(1-x)^d}{xd}.\end{aligned}$$

Hence, $S_d = dG_d(1) = \int_0^1\frac{1-(1-x)^d}{x}dx$. The change of variable $u = 1-x$ yields

$$S_d = \int_0^1 \frac{1-u^d}{1-u}du = \int_0^1 \sum_{k=0}^{d-1} u^k du = \sum_{k=0}^{d-1} \frac{1}{k+1} = H_d,$$

where $H_d = \sum_{k=1}^{d} \frac{1}{k}$ is the d-th harmonic number.

Finally, putting everything together,

$$\mathbb{E}[C_d] \leqslant d + \frac{dH_d}{-\log(1-\alpha)} + \varepsilon,$$

using the fact that $H_d = \log d + \gamma + O(1/d)$ and $\log(1-\alpha) \approx -\alpha \approx \frac{-\pi^2}{N^2}$, we get

$$\mathbb{E}[C_d] = O\left(N^2 d \log d\right). \tag{2}$$

D Proof of Lemma 4

We will first show the link between the function F introduced in the proof of lemma 2 and the size on every dimension of the interval I, then we use this to compute the number of points removed from the interval at each step, and finally we use the random graph assumption to compute the expected number of isolated points generated at each step (this is an upper bound on the final number of isolated points since partial coalescence can happen).

If we couple two one-dimensional intervals of different sizes, we can notice that the decrease of their respective size after T steps is the same for both until the smallest one coalesce (becomes a single point). Hence, after T steps, an interval with initial size N has size M if and only if N is larger than M and a grid with initial size $(N-M)$ has coalesced.

Let $s(N,d,T,i)$ be the size in dimension i on interval of initial size N^d, after T moves,

$$\mathbb{P}(s(N,d,T,i) > M) = \mathbb{P}(C^{(N-M)} > T) = F(N-M,T) - F(N-M-1,T),$$

using Equation (1). We can therefore compute

$$\mathbb{E}(s(N,d,T,i)) = \sum_{M=1}^{N} \mathbb{P}(C^{(N-M)} > T) = F(N,T) - F(1,T) = F(N,T)$$

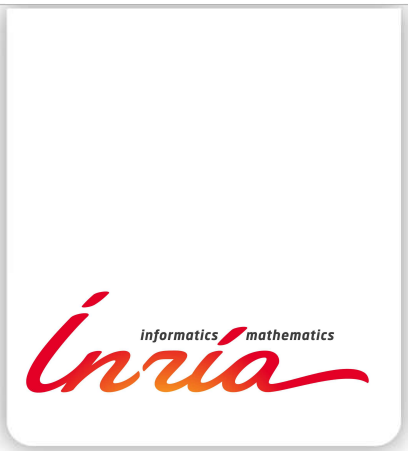
because $F(1,T) = 0$ by definition.

The surface $S(N,d,T)$ crossed for move T on direction d is either 0 (when on a border) or $\prod_{d' \neq d} s(N,d,T,d')$.

Using the independence of the size in the different dimensions of the state space, we have $\mathbb{E}[S(N,d,T)] = \mathbb{E}[s(N,1,T,1)^{d-1}] = F(N,T)^{d-1}$.

Hence the total number, Z of isolated points created over time, satisfies

$$\begin{aligned}
\mathbb{E}[Z] &= \mathbb{E}[\textstyle\sum_{T=0}^{\infty} S(N,d,T)\frac{k}{N^d}] = \frac{k}{N^d} \sum_{T=0}^{\infty} F(N, \frac{T}{d})^{d-1} \\
&\leq \frac{k}{N^d} \sum_{T=0}^{\infty} (N+1)^{d-1}(1-\alpha)^{\frac{d-1}{d}T} \\
&\leq \frac{k}{N} \left(\frac{(N+1)}{N}\right)^{d-1} \sum_{T=0}^{\infty} (1-\alpha)^{\frac{d-1}{d}T} \\
&\leq \frac{k}{N} \left((1 + \frac{d-1}{N} + O(\frac{d^2}{N^2})\right) \frac{1}{1-(1-\alpha)^{\frac{d-1}{d}}} \\
&\leq \frac{k}{N} \left((1 + \frac{d-1}{N})\right) \frac{(N+1)^2}{\pi^2} (1 + \frac{1}{d} + O(N^{-2})) \\
&\leq \frac{kN}{\pi^2} + O(kd + \frac{kN}{d}).
\end{aligned}$$



RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399